TP Graphes 6

Licence Informatique - 2nde année

Année 2023-2024

L'objectif de ce TP est d'expérimenter l'utilisation d'une bibliothèque graphique pour effectuer l'affichage d'un graphe et le plus court chemin entre 2 sommets dans celui-ci. Vous allez ainsi être amenés à compléter l'application développée dans le cadre du TP5 afin qu'elle puisse afficher un graphe dans une fenêtre graphique.

Préliminaire

Dans un premier temps, recopiez les fichiers `.cpp`, `.hpp` et le fichier `Makefile` de l'exercice 2 du TP précédent dans un dossier nommé `TP6`. Renommez ensuite le fichier `tp5.cpp` en `tp6.cpp`.

1 Ajout des positions

Pour pouvoir afficher chaque sommet à un emplacement de la fenêtre d'affichage, le format des fichiers représentant une matice d'adjacence a été modifié de la manières suivante : en début de chaque ligne figure un couple d'entiers qui représente les coordonnées auxquelles le sommet doit être affiché. Deux fichiers dans ce format vous sont fournis dans le dossier `Data` associé à cet énoncé (`grapheC01.txt` et `grapheC02.txt`) et seront utilisés dans le cadre de cet exercice.

Pour pouvoir tenir compte de ces nouvelles informations, on propose un nouveau type nommé `coordonnees`, défini comme suit :

```
/*
 * structure représentant une coordonnées cartesienne
 * dans une fenêtre d'affichage
 */
struct coordonnees {
  int x, y;
};
```

1. Ajoutez ce nouveau type à votre fichier `types.hpp`, avant la définition de la structure de matrice d'adjacence ;
2. Modifiez la structure `MatriceAdjacence` de la manière suivante :

   ```
   /*
    * structure représentant une matrice d'adjacence sous forme
    * de matrice creuse.
   */
   struct MatriceAdjacence {
     int ordre; // nombre de sommets du graphe
     Maillon* *lignes;// tableau à allouer de taille "ordre", représentant les lignes de la matrice
     coordonnees *positions;// tableau à allouer de taille "ordre", représentant la position
                            // du sommet dans la fenêtre d'affichage

   };
   ```

   où `positions` correspondra à un tableau contenant les coordonnées de chaque sommet de la matrice ;
3. Modifiez la fonction `creerMatrice` de manière à ce qu'elle alloue et initialise le tableau de coordonnées `positions` d'une matrice d'adjacence ;
4. Modifiez la fonction `effacerMatrice` pour qu'elle efface ce tableau ;

5. Modifiez la fonction `charger` pour qu'elle relise les coordonnées de chaque sommet et les range dans le tableau `positions` ;
6. Modifiez la fonction `afficher` afin qu'elle affiche aussi la position de chaque sommet devant sa ligne.

Vous testerez chaque étape de ces modifications avec les fichiers fournis.

2 Ajout du module graphique

L'enoncé de ce TP est fourni avec les fichiers `graphique.cpp` et `graphique.hpp`, qui seront utilisés pour effectuer les affichages graphiques. Après avoir récupéré ces fichiers :

1. ajoutez le module `graphique.cpp` à votre fichier `Makefile` de manière à ce qu'il soit considéré comme une dépendance de votre application ;
2. à la fin de la ligne d'éditions de liens, vous rajouterez `-lSDL2` qui précise au compilateur d'utiliser le code compilé de la librairie `SDL` qui est utilisée ici pour gérer le graphisme ;
3. à la fin de la ligne de compilation du module `graphique.cpp`, vous ajouterez le texte :
   `$(sdl2-config --cflags)`
   qui permettra au compilateur de trouver les différents fichiers de la librairie `SDL` nécessaires à la compilation de ce module ;
4. incluez le fichier `graphique.hpp` dans votre module principal ;
5. modifiez votre fonction `main` en y ajoutant le code suivant après l'affichage du chemin minimum dans la console et avant la suppression des tableaux utilisés dans l'algorithme de Dijkstra :

```
// ------ affichages graphiques -----

// création de la fenêtre
if(createWindow(mat)==EXIT_FAILURE){
  cout << "erreur de création de la fenêtre" << endl;
  return 0;
}

// affichage du graphe
drawGraph(mat);

// affichage des plus courts chemins
drawPaths(mat, parents);

// attendre l'appui sur une touche ou la fermeture de la fenêtre
waitForEnd();

// destruction de la fenêtre
destroyWindow();
```

6. Compilez votre application et testez là : une fenêtre graphique doit s'ouvrir, vide à ce stade. Pour quitter la fenêtre, vous pouvez soit appuyer sur une touche quelconque, soit cliquer sur la croix de fermeture de la fenêtre.

3 Dessin du graphe

Complétez la fonction `void drawGraph(MatriceAdjacence &m)` qui se trouve dans le fichier `graphique.cpp`, aux emplacements qui sont indiqués. Cette fonction a pour but de tracer le graphe, dont la matrice d'adjacence est fournie, dans la fenêtre. Vous disposez pour ce faire des deux fonctions graphiques suivantes :

— `void drawLine(coordonnees deb, coordonnees fin, SDL_Color c)` qui permet de tracer une ligne droite entre les points de coordonnées `deb` et `fin`, dans la couleur passée en troisième paramètre. Trois variables représentant les couleurs rouge, vert et bleu ont été définies dans le module, avec les noms `rouge`, `vert` et `bleu`.

— `void drawRect(coordonnees centre, int cote, SDL_Color col)` qui permet de tracer un rectangle dont le centre, le côté et la couleur sont fournies. On précise qu'une constante `TAILLE_RECT` a été définie dans le module, et pourra être utilisée pour préciser la valeur du paramètre `cote`.

Compilez et testez votre application.

4 Dessin des chemins les plus courts

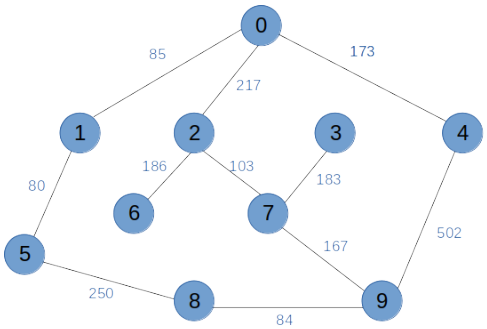
Après application de l'algorithme de Dijkstra, le tableau `parents` a été complété et permet de retrouver les chemins les plus courts entre chaque sommet et le sommet de départ. En vous inspirant que ce qui a été fait pour afficher les chemins trouvés dans la console (en mode texte), complétez les fonctions suivantes pour effectuer l'affichage graphique de ces chemins :

- `void drawPaths(MatriceAdjacence mat, int *parents)` : permet de lancer le tracé du chemin le plus court entre chaque sommet de graphe et le sommet de dpart ;
- `void drawPath(MatriceAdjacence mat, int sf, int *parents)` : permet de tracer **récursivement** le chemin entre le sommet d'indice `sf` et le sommet de départ.

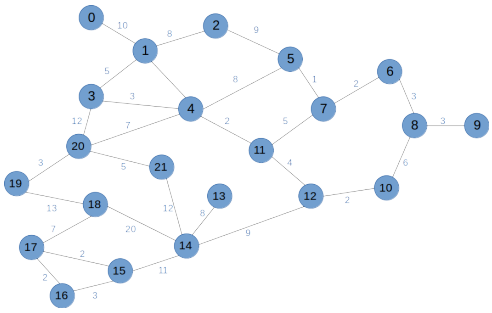
Il est conseillé d'utiliser une autre couleur que celle utilisée pour tracer les arêtes du graphes, afin de bien les distinguer de celles constituant le chemin le plus court.

5 Exemples

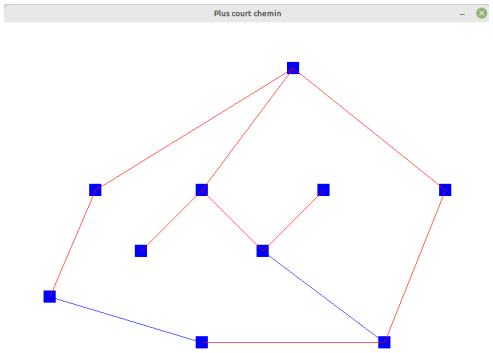
Les figures 1a et 1b illustrent la géométrie des deux graphes fournis et la numérotation des sommets utilisée, tandis que les figures 1c et 1d vous donnent un aperçu de l'affichage graphique à obtenir.



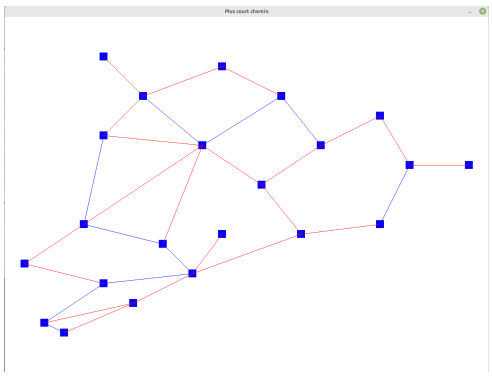
(a) Aperçu du graphe du fichier `grapheC01.txt`



(b) Aperçu du graphe du fichier `grapheC02.txt`



(c) Aperçu de l'affichage graphique des plus courts chemins dans le graphe `grapheC01.txt` (en rouge) à partir du sommet 0.



(d) Aperçu de l'affichage graphique des plus courts chemins dans le graphe `grapheC02.txt` (en rouge) à partir du sommet 4.

FIGURE 1 – Exemples d'affichage des plus courts chemins (en rouge) pour les deux graphes fournis ; les arêtes de couleur bleu correspondent à des liens existants mais sans faire partie d'un chemin le plus court.